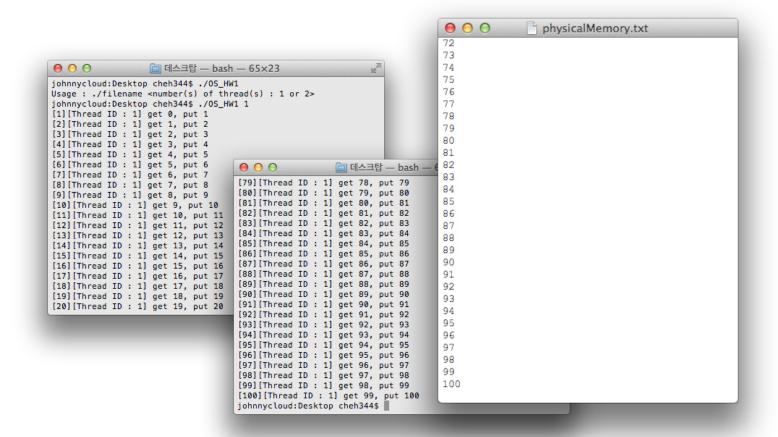
OS\_HW1 1/4

## #1. Race Condition

- 1. 다음 기능을 하는 프로그램을 만드시오
  - 1) 데이타 화일의 마지막 숫자를 읽어서 1을 더해서 그 화일의 끝에 추가해서 기록 (데이타 화일은 초기에 0 만 써있다.)
  - 2)위 함수를 100번 호출
  - 3)이 프로그램을 실행하면 0 부터 100까지 101개의 숫자가 기록될 것이다.
- 2. 이 프로그램을 두개 이상 동시에 실행하여 결과를 살펴보시오
  - 1) n개를 동시에 실행했을 때 화일의 마지막 숫자가 100\*n인가? 예: 2개를 동시에 실행했을 때 200
  - 2) 결과가 100\*n이 아니라면 그 원인은 무엇인지 설명하시오.

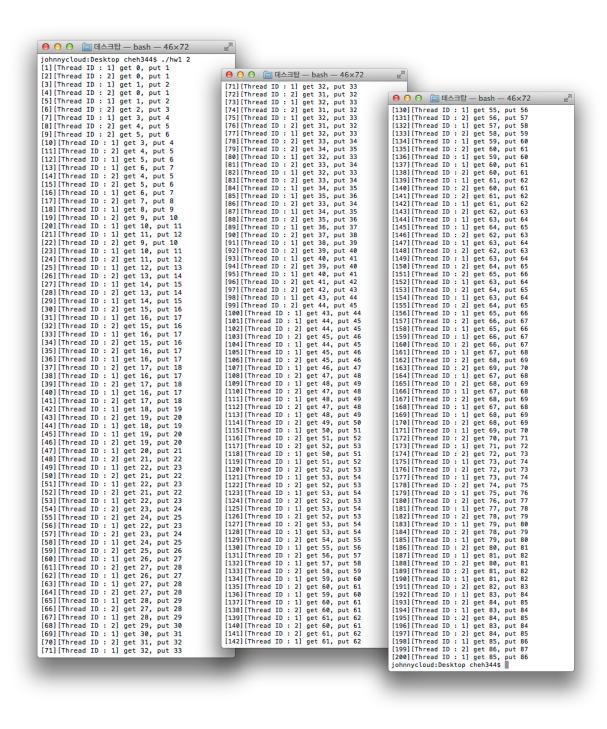
작성한 코드(OS\_HW1.c)는 argument로 스레드의 수(최대 2 개)를 입력받아, 그 수만큼의 간단한 스레드함수를 실행하는 프로그램입니다.

- 다음은 프로그램 실행예시입니다. 우선 한개의 스레드만 작동하는 조건입니다.



OS HW1 2/4

- 다음은 두개의 스레드를 작동하는 조건입니다.



## [원하는대로 값이 나오지 않았습니다.]

여기서 문제는 두 가지로 요약할 수 있습니다.

- I. 출력문에 숫자의 중복이 보인다.
- II. 결국 200번을 작동하는 동안, 이에 맞춰 마지막 숫자가 올라가지 않았다.

OS HW1 3/4

다음은 소스코드의 일부입니다. 다음 함수가 2개의 Thread로 동작하게 됩니다.

```
void * simpleThread (void *data) {
int id;
                                     * sleep()이 없을 시, 1번 스
char tmp[10];
                                      레드를 생성하고 나서, 2번
int memoryValue;
int i = 0;
                                      스레드를 생성하는 찰나의
id = *((int*)data);
                                      순간에 1번 스래드가 작업
                                      을 끝내버리는 상황이 발생
for (i = 0; i < 100; i++) {
                                      했습니다.
  sleep(1); ....
  if (i < 100) fseek(sharedMemory, -3, SEEK_END);</pre>
  else if (i < 1000) fseek(sharedMemory, -4, SEEK_END);</pre>
  else errHandling("value overflow");
  fscanf(sharedMemory, "%d", &memoryValue);
                                                           2
  fprintf(sharedMemory, "%d\n", memoryValue+1);
                                                           3
  printf("[%d][Thread ID : %d] get %d, put %d \n",
             ++runTimes, id, memoryValue, memoryValue+1);
return 0;
```

위 함수는 크게 세 영역으로 나눌 수 있습니다

- 1. 파일의 마지막 위치를 찾는다.
- 2. 마지막 위치의 숫자를 읽는다.
- 3. 마지막 숫자에 1을 더하여 다음 위치에 저장한다.

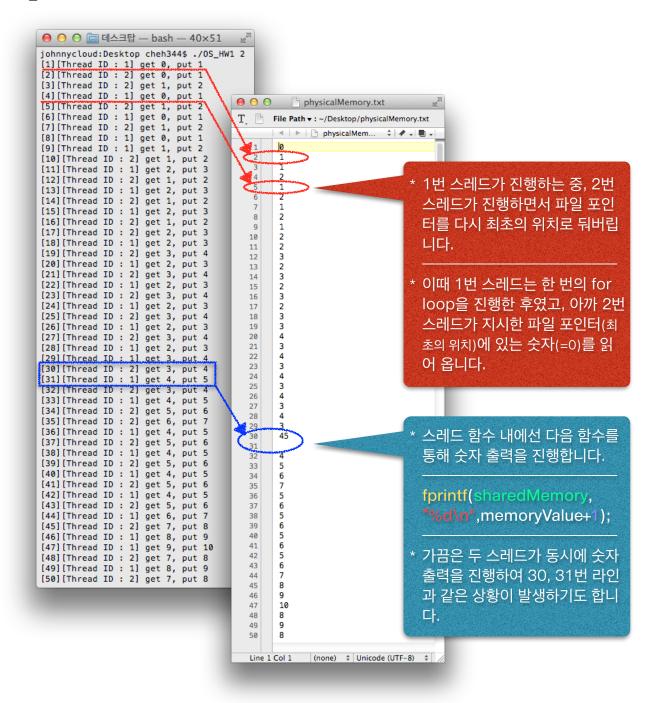
여기서 문제는 각각의 스레드가 따로 놀아, 서로를 제제하지 못해 다양한 문제가 발생하게 됩니다. 가령,

- 1번 스레드가 1번 영역을 실행.
- 동시에 2번 스레드가 1번 영역을 실행.
- 결국, 두 스레드가 같은 위치로 파일 포인터를 두게 된다.

더 나아가,

- 1번 스레드가 2번 영역을 진행한 직후, 2번 스레드가 꽤 많이 진행해 버리면, <u>1번 스레드</u>는 <u>'더는 마지막이 아</u> 닌 숫자'를 가져왔다고 볼 수 있다.
- 따라서 1번 스레드는 2번 스레드가 작성한 이후에 해당하는 '<u>마지막 위치'에 '더는 마지막이 아닌 숫자'를 저장</u> 하게 된다.
- 또한, 1번 스레드에 의해 위 과정이 진행되면서 '마지막 위치'와 '마지막 숫자'가 바뀌었으므로, <u>2번 스레드 또</u>한 위의 문제를 같이 겪게 된다.
- •이 문제가 반복되면서 결국 숫자의 최종값이 원하는 값(100\*n)에 결코 다다를 수 없게 된다.

OS HW1 4/4



위의 문제를 해결하기 위해선 위 세 영역을 Critical-Section 으로 두어, 한 스레드가 위 영역에 접근했을 시 다른 스레드의 접근을 대기시켜 주어야 할 것입니다.